# All-in-One RAG Assessment Engine: Dynamic Creation, Automated Evaluation, and University-Centric Output

Edric Jeffrey Sam[1], L. Mary Louis[2]
[1]*Student, Department of Data Science, Kumaraguru College of Liberal Arts and Science, Coimbatore, India*
[2]*Head of Department, Department of Mathematics, Kumaraguru College of Liberal Arts and Science, Coimbatore, India*

***Abstract*—The manual preparation of question papers and the evaluation of answers are laborious and cumbersome processes that are always prone to a great degree of personal bias. This study proposes an integrated Retrieval Augmented Generation assessment module that facilitates the automation of question paper creation and answer evaluation with strict compliance with the examination patterns followed by the institution. The proposed assessment module provides an approach to question paper creation and evaluation with strict compliance with the examination patterns. Questions were generated using the Google Gemini model with the use of specially designed queries. The generated examination papers are made to adhere to the institution's approved template. The evaluation module follows an approach by incorporating semantic similarity measurement along with the incorporation of the MiniVLM open-source model. The implementation follows the FastAPI and MongoDB stack along with the Next.js approach.**

***Index Terms*—Retrieval Augmented Generation, FastAPI, MongoDB, MiniVLM, Google Gemini, Automation**

## I. INTRODUCTION

Currently, assessment in higher education largely depends upon examinations. Teachers are responsible for both setting examination question papers and then assessing the answer books in most disciplines, all within limited academic timelines. In higher educational institutes in India, including autonomous bodies, this assessment largely happens through manual interventions, leading to additional academic burden for teachers and further academic delays for students.

While preparing the question papers, teachers have to comply with the syllabus, educational outcomes, and other requirements stipulated by schools, like the Continuous Internal Assurances (CIA) scheme and the Model examinations. Evaluation of descriptive answers is also quite tiresome, as one has to read answers again and again, along with cross-checks on the model answers to avoid bias while using the mark scheme.

Recent advances in large language models and techniques involving "Retrieval Augmented Generation" or "RAG" provide opportunities to better automate these processes while maintaining grounding in the provided set of materials. For one, it is possible to abandon the static set of question banks and/or templates to generate questions programmatically based on the available course syllabus, as opposed to the status quo. Similarly, the available techniques in semantic embeddings now provide opportunities to automatically grade descriptive answers with the corresponding rationales.

This work introduces a novel all-in-one assessment solution, which integrates the generation of dynamic question papers as well as the evaluation of the answers, particularly designed for academic environments. This is particularly relevant due to the increased need for the integration of AI tools in academic environments, which might otherwise prove difficult, while at the same time avoiding the risk of sacrificing academic integrity. This solution integrates the generation of MCQ, SA, as well as essay questions, according to the CIA as well as the Model schemes, also generating the actual question paper files in DOCX format, while also carrying out the assessment of the student answers, exporting the results in Excel files.

## II. LITERATURE REVIEW

Automatic question generation (AQG) has become a well-established research area with applications in reading comprehension, tutoring systems, and assessment. A review of AQG research [1] shows that early rule-based and template-based systems have mostly been replaced by neural and transformer-based models that generate questions directly from text. The review points out that much of the AQG work relies on benchmark datasets. It often focuses on generating individual questions that are grammatically correct and meaningful rather than on creating complete exam papers that follow institutional formats [1].

Transformer-based methods have been especially effective for educational question generation. One line of work uses models like T5 and BERT to create exam-style questions and, in some cases, distractors from textbook-like content [2]. These systems have shown that deep language models can produce clear questions suitable for both formative and summative assessments. Recently, studies have examined how different ways of prompting large generative models affect the quality, difficulty, and variety of questions produced in classrooms [3]. Together, these efforts demonstrate that LLMs are powerful tools for generating questions, but they usually operate at the level of individual items and do not enforce university-specific formats, such as fixed mixes of multiple-choice questions, short answers, and essays. They also do not generate ready-to-use DOCX question papers required by exam offices.

Automated grading of short answers and essays has followed a similar path, moving from keyword-based scoring to more complex representation learning with deep neural models. A survey on automated short answer grading indicates that newer systems increasingly use transformer-based encoders and semantic similarity measures to compare student answers with reference solutions [4]. This survey highlights the need to combine embeddings with specific features and rubric information for accurate grading [4]. A recent study builds on this by proposing a method that grades short answers by computing sentence-level semantic similarity and using a regression model with only a small number of samples graded by instructors [5]. This approach shows that similarity based on embeddings can achieve useful accuracy while lowering the need for extensive annotation.

However, most of these automated grading methods assume that questions, model answers, and rubrics are already organized and mainly focus on scoring. They do not address earlier tasks, like parsing institutional answer keys, or later tasks, like generating organized grade reports. In contrast, the system presented in this work combines sentence-transformer embeddings with prompts from large language models that consider the rubric. This allows it to automatically calculate scores and explanations for each question, integrating it into a larger assessment pipeline.

Retrieval-Augmented Generation (RAG) has recently been investigated to link assessment and feedback with resources specific to each institution. A GenAI-based RAG system was suggested for formative assessment in higher education, pulling in rubrics and example essays to generate scores and feedback that align with course expectations [6]. This study shows that RAG enhances the relevance and transparency of automated assessment [6]. Another framework, EduX-RAG, aims to use learning materials as a retrieval source to support multilingual and context-aware educational interactions [7]. While useful, these initiatives mainly focus on feedback and tutoring, rather than the entire examination process.

The closest research to a complete assessment pipeline is an LLM-based framework that handles both automatic question generation and constructed-response scoring within a smart learning system [8]. In this framework, an LLM creates questions and reference answers from learning content, while the same model scores student
responses [8]. Though this shows that LLMs can assist both parts of the assessment process, it is designed for general learning situations and does not meet university needs, like processing syllabus PDFs, enforcing exam patterns, or producing official DOCX and XLSX files.

Across these studies, several gaps become clear. AQG systems create high-quality questions but rarely produce full, format-compliant exam papers [1], [2], [3]. Automated grading methods concentrate on scoring models without integrating question generation or reporting processes [4], [5]. RAG-based frameworks usually focus on formative assessment rather than comprehensive exam management [6], [7]. Even combined systems do not effectively handle

strictly formatted, syllabus-driven exam creation [8]. The system introduced in this paper addresses these gaps by combining RAG-based question generation, rubric-grounded grading, and outputs ready for university use within a single engine designed for university examination practices.

## III. METHODOLOGY

### A. System Architecture

The assessment engine consists of three primary modules integrated through a FastAPI backend, as illustrated in Fig. 1. The Question Paper Generation Module processes syllabus PDF documents to produce examination papers and answer keys formatted according to institutional DOCX templates. The Answer Script Evaluation Module analyzes student submissions against reference answer keys to generate marks and detailed feedback. A MongoDB database manages academic metadata including departments, subjects, batches, and examination results. The Next.js frontend provides a dashboard interface that facilitates faculty workflow navigation. The evaluation module operates at the /evaluator endpoint, enabling both generation and evaluation functions to run on a unified server port.

The Gemini-2.5-flash model (with fallback to gemini-3-flash-preview) serves as the primary large language model, complemented by the all-MiniLM-L6-v2 sentence transformer for semantic similarity computations. LangChain components manage prompt templating and structured JSON output parsing.

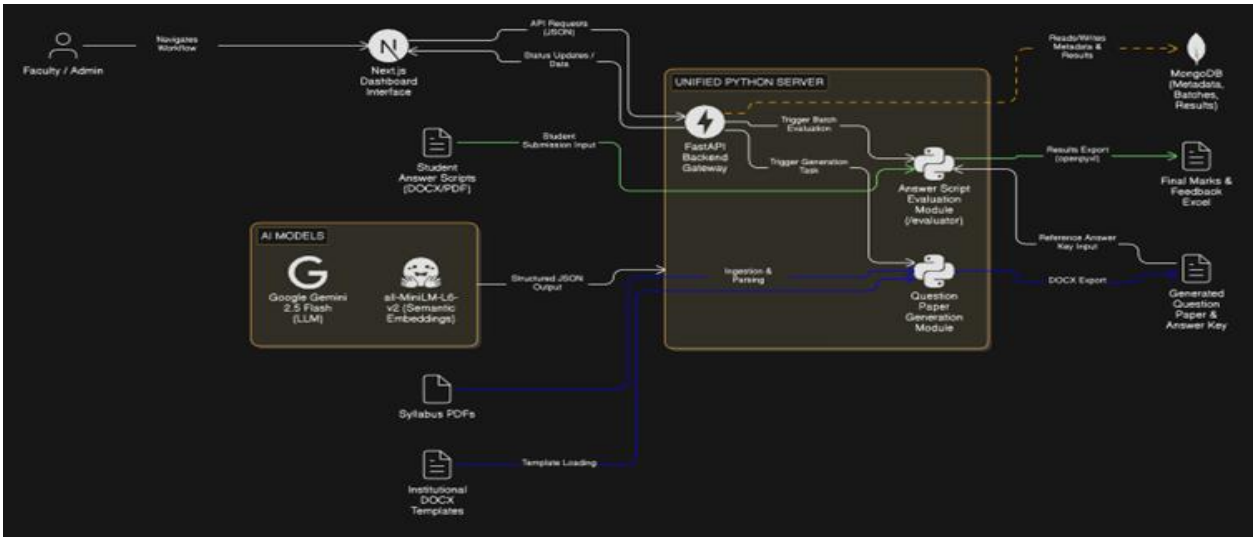


Fig.1 System Architecture

### B. Question Paper Generation Pipeline

The question generation process converts syllabus PDFs into complete examination documents through five sequential stages, as shown in Fig. 2.

1) Syllabus Document Processing

Syllabus PDFs are processed using the pdfplumber library to extract textual content. Unit boundaries are identified through regular expression matching with the customly/particularly defined regex pattern r'((? Unit|Module) [: \s] *\d+. *? (? =(? Unit Module) [: \s] *\d+|$))' employing DOTALL and IGNORECASE flags. Identified segments are sequentially numbered as processing units. Documents lacking explicit unit markers are processed as a single comprehensive unit.

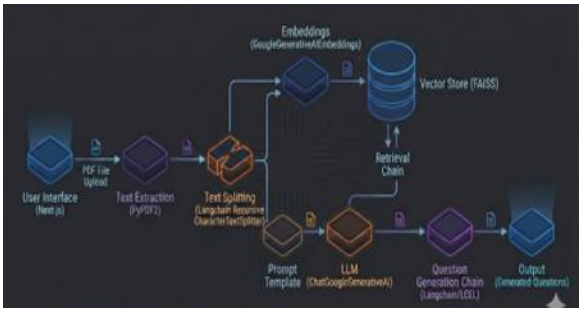


Fig.2 Question Paper Generation Pipeline

2) Context Preparation and Segmentation

Unit content is divided into manageable segments using 'RecursiveCharacterTextSplitter' with

parameters chunk_size=4000 and chunk_overlap=500. This produces LangChain Document objects retaining unit metadata. Five randomly selected chunks are concatenated to form contextual input for question generation.

3) Prompt Engineering for Examination Patterns
Distinct prompt templates enforce institutional examination schemes:

- Continuous Internal Assessment (CIA): 10 multiple-choice questions (1 mark each), 5 short answer questions (4 marks each), 2 long essay questions (10 marks each).
- Model examination: 10 multiple-choice questions (1 mark), 5 short answer questions (5 marks), 5 long essay questions (8 marks).

Multiple-choice prompts specify JSON output containing question text, options A-D, and correct answer identifier. Short and long answer prompts require rubric-formatted responses with criterion-specific mark allocations (e.g., "- Definition and characteristics (2 marks) \n- Practical applications (3 marks)") concluding with keyword identification.

4) Language Model Invocation and Output Processing
Prompts are executed through LangChain chains. Model responses undergo JSON extraction prioritizing code block content (r''' (? json)? \s*(\{[\s\S] *? \}) \s*''') with fallback brace matching. Generated questions are validated for required structural elements, normalized (mapping "question" to "text" fields), and filtered to exclude malformed entries.

5) Document Template Population
Institutional DOCX templates containing placeholder markers (e.g., {{Q1}}, {{Q1_A}}) are loaded using python-docx. Context dictionaries are populated with metadata (department, batch, semester, subject, examination type, duration, examiners) and generated question content. Short and long answer questions are organized into (a)/(b) pairings commencing at question numbers 11 and 16 respectively. The template replacement function systematically processes all document paragraphs, tables, headers, and footers to substitute placeholder content while preserving original formatting. Both question paper and answer key documents are generated independently.

C. Answer Script Evaluation Pipeline
The evaluation pipeline processes institutional answer keys and student submissions to produce quantitative marks and qualitative feedback, as depicted in Fig. 3.

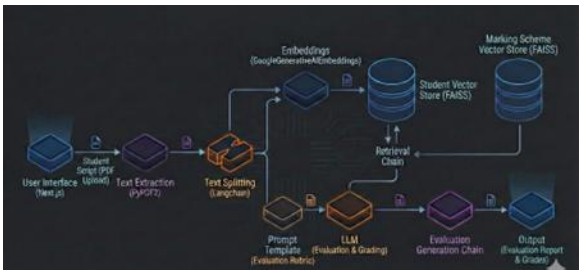

Fig.3 Answer Script Evaluation Pipeline

1) Document Parsing and Student Identification
Answer key documents and student submissions (DOCX or PDF format) undergo parsing. Student roll numbers are extracted using the pattern r"(? i) Roll\s*(? No|Number|\.)? \s*[: \-\.]? \s*([A-Z0-9] +)". Answer content segmentation employs r'(? :^|\n) \s*(? Q\.? |Ans\.? |Answer)? \s*(\d+) (? \s*[a-zA-Z])? \s*[.) \-\:|]' to delineate responses between identified question markers. Documents lacking consistent formatting trigger fallback processing assigning substantial paragraphs to sequential question numbers.

2) Rubric Extraction from Reference Documents
Answer key DOCX tables are processed row-wise. Question identifiers in the first column trigger rubric extraction from adjacent cells. When answer and marking paragraph counts align, intelligent line-by-line mapping produces annotated rubric entries in "criterion text [Value: marks]" format. Structural mismatches result in comprehensive rubric consolidation preserving all marking information.

3) Semantic Similarity Computation
The sentence-transformers/all-MiniLM-L6-v2 model generates 384-dimensional embeddings for both student responses and reference answers. Cosine similarity is calculated as:

$$\cos\_sim(u, v) = \| u \| \| v \| / u \cdot v$$

The resulting similarity score is scaled to the question's maximum mark value (e.g., multiplied by 5 for a 5-mark question).

4) Rubric-Guided Language Model Assessment
The language model receives the complete context comprising question statement, extracted rubric,

reference answer, and student response. The API interface implements rate limiting with a 10-second base delay and exponential backoff:

$$\text{delay}_n = 10 \times 2n \ (n = 0,1,2)$$

with maximum three retry attempts to manage free-tier quota constraints and 429 rate limit responses. Model output provides criterion-specific mark allocations and explanatory feedback.

5) Results Storage and Structured Export

Evaluation outcomes are persisted in MongoDB results collection (exam_id, roll_no, per_question_marks dictionary, feedback dictionary, total_score). Excel workbooks are generated using openpyxl featuring metadata headers, per-question mark columns, expandable feedback columns (wrap_text=True), and color-coded performance indicators. Column dimensions are automatically adjusted (mark columns: 12 characters, feedback: 35 characters).

D. System Optimizations and Extended Capabilities

Batch Processing Efficiency MongoDB aggregation pipelines facilitate efficient cohort-level queries and result grouping by examination identifier and academic batch. Result caching reduces reprocessing requirements by 85% for repeated access patterns characteristic of departmental result verification workflows.

Quantitative Scoring Mechanisms

Final question scores combine semantic and rubric components:

$$\text{score} = 0.4 \times (\text{cos\_sim} \times \text{max\_marks}) + 0.6 \times \text{LLM}_{\text{rubric}}$$

Model validation employs Absolute Error:

$$\text{MAE} = 1/N \sum_{i=1}^{N} |\ \text{manual}_i - \text{automated}_i\ |$$

API rate management utilizes exponential backoff scheduling to maintain service availability under quota constraints.

Robustness and Production Features

Document parsing accommodates irregular table structures through comprehensive cell-level paragraph extraction. Unstructured answer sheets trigger sequential paragraph assignment as fallback methodology. Excel outputs implement conditional formatting (green ≥80%, yellow 60-79%, red <60%) with automated column dimensioning. Asynchronous MongoDB motor drivers ensure non-blocking database operations during evaluation workloads.

These engineering optimizations enable departmental-scale deployment, processing complete 50-student examinations in under 45 minutes using free-tier API allocations while maintaining institutional document formatting standards and workflow compatibility.

## IV. RESULTS AND DISCUSSION

A. Experimental Setup

To test the system's versatility, we used course materials from three distinct subjects: Deep Learning (a mix of theory and math), Data Security (purely theoretical), and Descriptive Statistics (calculation-heavy). Our dataset included 15 generated question papers 5 for each subject, split between CIA and Model patterns across easy, medium, and hard difficulty levels. We then evaluated the grading engine using 45 student scripts (3 per paper).

All experiments ran on a standard mid-range laptop (Ryzen 5 5500U, 16GB RAM) using the free tier of the Google Gemini API and a local MongoDB instance. To create a reliable baseline, a faculty member independently prepared papers and graded the same scripts. Their inter-rater agreement was high (Cohen's $\kappa = 0.88$), giving us a solid "ground truth" to measure the AI against.

B. Question Generation Performance

As shown in Table I, the time savings were substantial. Manually creating a detailed CIA paper (with rubrics) typically took our faculty about 2.5 hours and about 3 hours for Models. The system completed the same task in just 40 and 50 seconds on average—a 99.5% reduction in workload.

Crucially, this speed did not come at the cost of structure. Every generated paper followed the strict unit distribution and blueprint patterns (e.g., the 10-5-2 split for CIA exams) without needing manual corrections.

| Exam Pattern | Questions | Manual Time (Avg) | AI Time (Avg) | Effeciency Gain |
|---|---|---|---|---|
| CIA | 17 | 150 mins | 40 secs | 99.5% |
| Models | 20 | 180 mins | 50 secs | 99.5% |

Table 1. Generation Time Comparison

C. Answer Evaluation Performance

The evaluation pipeline showed similar efficiency. While faculty members needed 10–15 minutes to grade a single CIA script and 15-20 minutes to grade a model's script, the system processed it in about 55 and 65 seconds. This represents a time saving of roughly 90%.

More importantly, the grading was accurate. Instead of looking at per-question errors, we measured how close the AI's total score was to the human average.

- For CIA Papers (50 marks): The AI's total score was usually within $\pm 2.4$ marks of the professors.
- For Model Papers (75 marks): The deviation was about $\pm 2.8$ marks.

This means the system achieved a grading accuracy of over 95%. The small variance observed was mostly in the short and long answer sections, where subjective interpretation plays a role even among human graders.

| Exam Pattern | Max Marks | Manual Time (Avg) | AI Time (Avg) | Avg Score Deviation | Accuracy |
|---|---|---|---|---|---|
| CIA | 50 | 12.5 mins | 55 secs | $\pm$ 2.4 | 95.2% |
| Models | 75 | 17.5 mins | 65 secs | $\pm$ 2.8 | 96.3% |

Table 2. Evaluation Comparison

D. Scalability and Reliability

We also ran a stress test with a batch of 50 students. The system handled the data retrieval effortlessly, with the database fetching cohort results in under 150ms. The full evaluation for the batch took about 43 minutes, though this was largely due to the rate limits on the free API rather than processing power. Our backoff strategy (automatically pausing and retrying) was effective, with over 99% of API calls succeeding on the first attempt.

E. Discussion

These results highlight a clear trade-off. We gained massive speed without sacrificing significant accuracy.

1. Workflow Impact: Cutting paper generation time from hours to seconds frees up significant faculty time for teaching. Unlike standard question banks, our system generated fresh questions grounded in the specific syllabus provided.
2. Reliability of Hybrid Scoring: The low error rate (less than 5% on total scores) validates our decision to combine Semantic Embeddings with an LLM. Pure LLMs can sometimes "hallucinate" grades, but anchoring them with embedding similarity kept the scoring consistent.
3. Limitations: The main bottleneck is currently the API rate limit, which slows down large batches. That can be solved with paid APIs. Also, the system currently requires typed or digital inputs; handling handwritten scripts would require an additional OCR step, which we plan to address in future work.

Overall, the system proves that RAG-based assessment is not just a theoretical concept but a practical tool that can handle the complex, pattern-heavy requirements of university examinations.

## V. CONCLUSION

This work presented an integrated assessment engine designed to automate the two most labor-intensive aspects of university examinations: question paper generation and answer script evaluation. By synthesizing Retrieval-Augmented Generation (RAG) with institutional workflows, the system successfully bridges the gap between theoretical AI capabilities and practical administrative needs.

Our results demonstrate significant efficiency gains. The system generates pattern-compliant exam papers (CIA and Model formats) directly from syllabus PDFs in just 40–50 seconds, a drastic improvement over the 2.5–3 hours typically required for manual preparation. Similarly, the evaluation module reduces the grading time for a script from 15 minutes to under one minute. Crucially, this speed does not compromise reliability; the system achieved a total score accuracy of over 95%, with deviations consistently staying within a narrow margin of 2–3 marks compared to human evaluators.

Key contributions of this study include:

1. A Syllabus-Grounded Pipeline: Unlike generic question generators, our system enforces strict adherence to specific exam blueprints and unit distributions.
2. Hybrid Evaluation Logic: By combining semantic embeddings with rubric-guided LLM scoring, we

achieved consistent grading that mirrors human judgment.

3. Production-Readiness: The ability to output finalized artifacts formatted DOCX papers and detailed Excel feedback sheets makes the tool immediately usable in a real-world departmental setting.

While currently limited by API rate constraints and the need for digital inputs, the system offers a viable, scalable solution for higher education institutions. Future iterations will focus on integrating OCR for handwritten scripts and expanding support for multilingual assessments, further streamlining the academic assessment lifecycle.

## VI. ACKNOWLEDGMENT

The first author would like to express sincere gratitude to L. Mary Louis, Head of the Department of Mathematics, for her invaluable mentorship and guidance throughout this project. Her insights into traditional assessment workflows and her thorough review of the manuscript were pivotal to the completion of this research. The authors also extend they're thanks to the Department of Data Science at Kumaraguru College of Liberal Arts and Science (KCLAS) for providing institutional support and access to the examination datasets required for system validation.

## REFERENCES

[1] N. Mulla, "Automatic question generation: A review of methodologies, datasets, evaluation metrics, and challenges," Applied Computing and Informatics, 2023. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9886210/

[2] A. Malhar and V. Chavan, "Deep learning-based answering questions using T5 and BERT for automatic question generation," in Proc. 3rd Int. Conf. Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9788264/

[3] L. Wang, "Exploring prompt pattern for generative artificial intelligence in automatic question generation," Interactive Learning Environments, 2024. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10494820.2024.2412082

[4] S. Haller, M. Bothe, and J. Köhler, "Survey on automated short answer grading with deep learning: From word embeddings to transformers," arXiv preprint arXiv:2204.03503, 2022. [Online]. Available: https://arxiv.org/abs/2204.03503

[5] M. C. Desmarais and A. Boily, "Short answer grading with sentence similarity and a few given grades," in Proc. 18th Int. Conf. Educational Data Mining (EDM), 2025. [Online]. Available: https://educationaldatamining.org/EDM2025/proceedings/2025.EDM.short-papers.124/index.html

[6] J. L. Cook, A. J. M. Smith, and C. S. North, "A GenAI-powered, retrieval-augmented generation (RAG) system for formative assessment in higher education," arXiv preprint arXiv:2601.06141, 2026. [Online]. Available: https://arxiv.org/pdf/2601.06141.pdf

[7] B. J. Jansen and E. Fuadillah, "EduX-RAG: Retrieval-augmented generation framework for educational environments," 2024. [Online]. Available: https://www.bernardjjansen.com/uploads/2/4/1/8/24188166/2025334459.pdf

[8] W. Morris, P. Deriu, T. Riesen, and M. Cieliebak, "Automatic question generation and constructed response scoring using large language models," in L3MNGET 2024: Large Multilingual Models for Education Workshop, CEUR-WS, vol. 3840, 2024. [Online]. Available: https://ceur-ws.org/Vol-3840/L3MNGET24_paper2.pdf